1. Introduction and Motivation
2. Principal Component Analysis (PCA)
3. Independent Component Analysis (ICA)
4. Non-negative Matrix Factorization (NMF)
5. Application and Result
6. Conclusion
7. Future Work
8. Code
9. Poster

Introduction and Motivation
The Introduction and Motivation of the Project

Introduction and Motivation

Introduction and Background

In real life, often, we come across data that doesn't have outputs (labels). In these cases, we strive to find the hidden structure and latent groupings within the data. This is when unsupervised machine learning comes into play. Popular unsupervised learning techniques include K-means, SOM (self-organizing map), various clustering algorithms (spectral clustering, hierarchical clustering) and graphical models. Here, we discuss unsupervised machine learning within the context of various matrix factorizations. The three algorithms we chose are **Independent Component Analysis** (**ICA**), **Principal Component Analysis** (**PCA**) and **Nonnegative Matrix Factorization** (**NMF**). They are compared using two applications of unsupervised machine learning in real-life scenario: the **cocktail party problem** and the **handwritten digit recognition**.

Motivation

Let's assume we have a data matrix of size nxp, where n corresponds to n iid observations and p corresponds to p features; very often, p is very large and the data is noisy. What we want is to find the relationship among the variables and uncover the major patterns underlying the data. What matrix factorization does is to reduce the dimensionality – It projects the data matrix into a lower dimension space so what seems random and messy in the high dimension becomes structured and grouped in lower dimension.

Principal Component Analysis (PCA)
A brief discussion of PCA.

Principal Component Analysis

PCA is essentially just SVD. The only difference is that we usually center the data first using some grand mean before doing SVD. There are three perspectives of views for PCA. Each of them gives different insight on what PCA does.

Low-rank Approximation

Equation:

$$\min_{Z} \quad \frac{1}{2}\left|\left|X - Z\right|\right|_F^2$$
$$subject\,to \quad rank(Z) \leq K$$

where Frobenius norm is a matrix version of sums of squared. This gives the interpretation of dimension reduction. Solution to the problem is:
$Z = \sum_{i=1}^{K} U_k d_k V_K^T$

We do lose some information when doing dimension reduction, but the majority of variance is explained in the lower-rank matrix (The eigenvalues give us information about how significant the eigenvector is. So we put the eigenvalues in the order of the magnitude of the eigenvectors, and discard the smallest several since the contribution of components along that particular eigenvector is less significant comparing that with a large eigenvalue). PCA guarantees the best rank-K approximation to X. The tuning parameter K can be either chosen by cross-validation or AIC/BIC. This property is useful for data visualization when the data is high dimensional.

Matrix Factorization

Equation:

$$\underset{U,D,V}{\text{minimize}} \left\{ \frac{1}{2} \| X - UDV^T \|_F^2 \right\}$$
$$subject to\ U^T U = I, V^T V = I, D \in diag^+$$

This gives the interpretation of pattern recognition. The first column of U gives the first major pattern in sample (row) space while the first column of V gives the first major pattern in feature space. This property is also useful in recommender systems (a lot of the popular algorithms in collaborative filtering like SVD++, bias-SVD etc. are based upon this “projection-to-find-major-pattern” idea).

Covariance

Equation:

$$\begin{aligned} \max \quad & V_K^T X^T X V_K \\ subject\ to \quad & V_K^T V_K = 1, V_K^T V_j = 0 \end{aligned}$$

$$X^T X$$

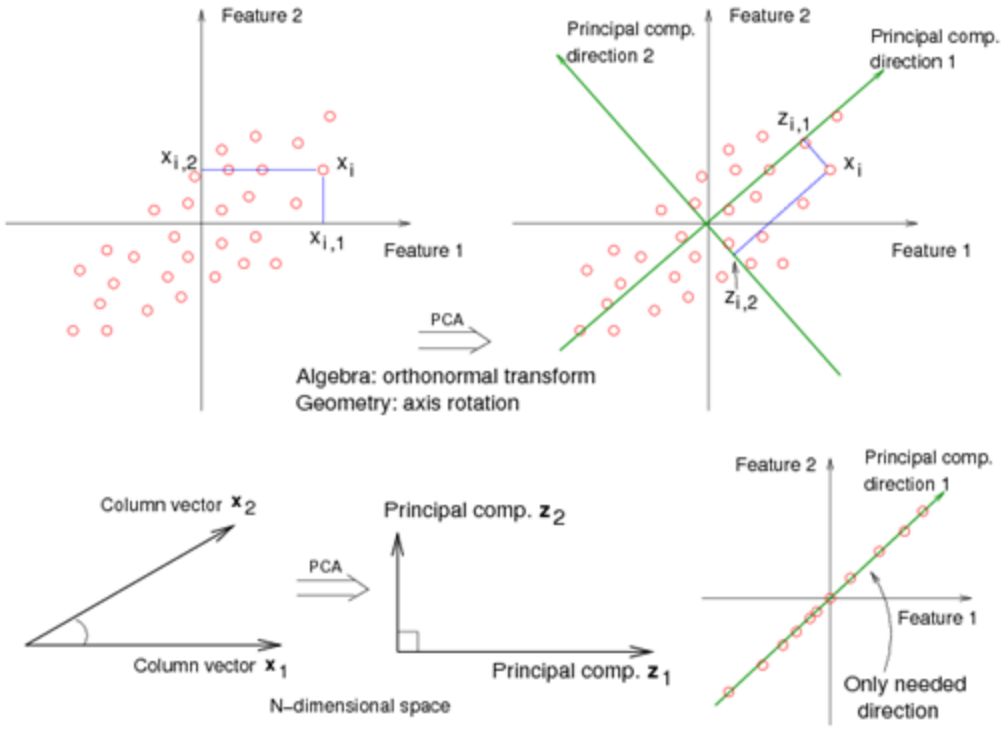
here behaves like covariates for multivariate Gaussian. This is essentially an eigenvalue problem of covariance:

$$X^T X \ = \ V D^2 V^T$$

and

$$X X^T \ = \ U D^2 U^T$$

. Interpretation here is that we are maximizing the covariates in column and row space.



(Figure Credit: https://onlinecourses.science.psu.edu/stat857/node/35)

The Intuition Behind PCA

The intuition behind PCA is as follows: The First PC (Principal Component) finds the linear combinations of variables that correspond to the direction with maximal sample variance (the major pattern of the dataset, the most spread out direction). Succeeding PCs then goes on to find direction that gives highest variance under the constraint of it being orthogonal (uncorrelated) to preceding ones. Geometrically, what we are doing is basically a coordinate transformation – the newly formed axes correspond to the newly constructed linear combination of variables. The number of the newly formed coordinate axes (variables) is usually much lower than the number of axes (variables) in the original dataset, but it's still explaining most of the variance present in the data.

Another Interesting Insight

Another interesting insight on PCA is provided by considering its relationship to Ridge Regression (L2 penalty). The result given by Ridge Regression can be written like this:

Equation:

$$\widehat{Y} = X\widehat{\beta}^r = \sum_{j=1}^{p} u_j \frac{d_j^2}{d_j^2 + \lambda} {u_j}^T y$$

The term in the middle here, $\frac{d_j^2}{d_j^2+\lambda}$, shrinks the singular values. For those major patterns with large singular values, lambda has little effect for shrinking; but for those with small singular values, lambda has huge effect to shrink them towards zero (not exactly zero, unlike lasso - L1 penalty, which does feature selection). This non-uniform shrinkage thus has a grouping effect. This is why Ridge Regression is often used when features are strongly correlated (it only captures orthogonal major pattern). PCA is really easy to implement - feed the data matrix(n*p) to the SVD command in Matlab, extract the PC loading(V) and PC score(U) vector and we will get the major pattern we want.

Independent Component Analysis (ICA)
A brief discussion of ICA (Independent Component Analysis).

Independent Component Analysis

Imagine the data we have is a linear mixture of unknown latent factors and the factors are mutually independent and non-Gaussian, our goal is to find the underlying structure and identify these components. As shown before, PCA assumes Gaussianity. The condition

and

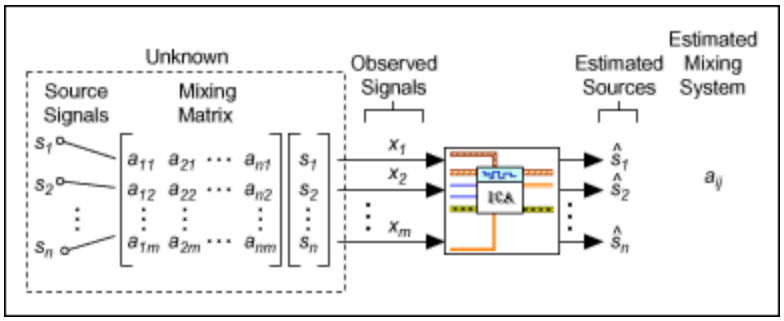
imply independence only when the data is Gaussian. ICA corrects this by looking for maximally independent components (rather than uncorrelated) ones. We know that uncorrlation is characterized by:

while independence is given by

The independence is stronger than uncorrelation because it measures the existence of any relationship. On the other hand, uncorrlation only measures linear relationship. Model for ICA looks like this:
Equation:

where A is the mixing matrix and S is the source signals (rows of S independent) and we recover the signal using the “whitening” matrix W, where:

Popular algorithms do this by maximizing its distance from Gaussian using either entropy or neg-entropy (Gaussian has maximal entropy) – they are solved using quasi-Newton scheme. In addition to this, some often used contrast functions to Gaussian include 3rd moment skewness, 4th moment kurtosis (kurtosis is zero for a Gaussian random variable) and sigmoidal. We used FastICA. It's efficient and converges quickly.



(Figure Credit: http://zone.ni.com/reference/en-XX/help/372656B-01/lvasptconcepts/tsa_multivariate_stat_analysis/)

Non-negative Matrix Factorization (NMF)
A brief discussion of NMF (Non-negative Matrix Factorization).

Non-negative Matrix Factorization

NMF provides an alternative approach to decomposition that assumes that the data and the components are non-negative. The model for continuous data looks like this:

$$\min_{W,H} \frac{1}{2} \|X - WH\|_2^2$$
$$subject\ to\ W_{ij} \geq 0,\ H_{ij} \geq 0$$

There are two views for NMF, and they provide different insights.

Archetypal Analysis

PCA tends to group both positively correlated and negatively correlated components together (it's only looking for variables with strong correlation). On the other hand, NMF, by forcing W and H to be positive, finds patterns with the same direction of correlation.

Fuzzy Clustering

W is sparse and it gives probabilistic cluster memberships (columns of W corresponds to kth cluster) and columns of H gives variables that define the kth cluster.

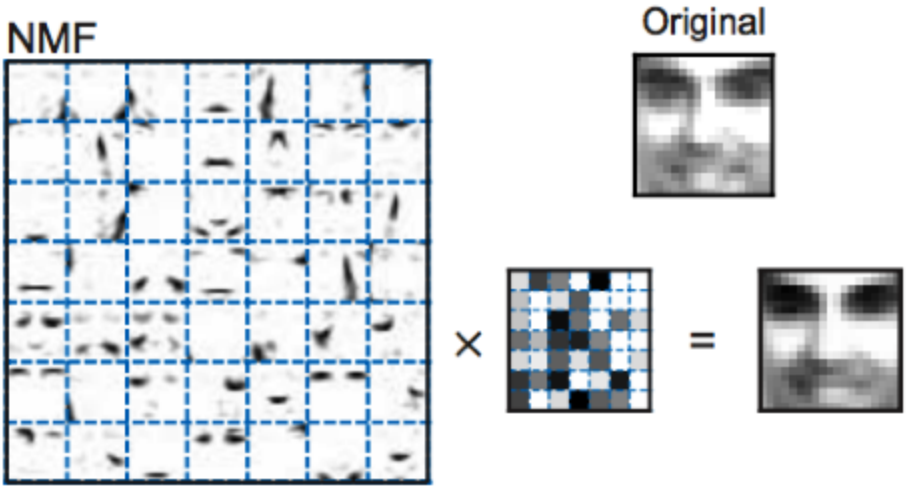
For count data, the model looks slightly different (related to Poisson):
Equation:

$$\underset{W,H}{\text{maximize}} \sum_{i=1}^{n} \sum_{j=1}^{p} [X_{ij} \log(W_i H_j) - W_i H_j]$$

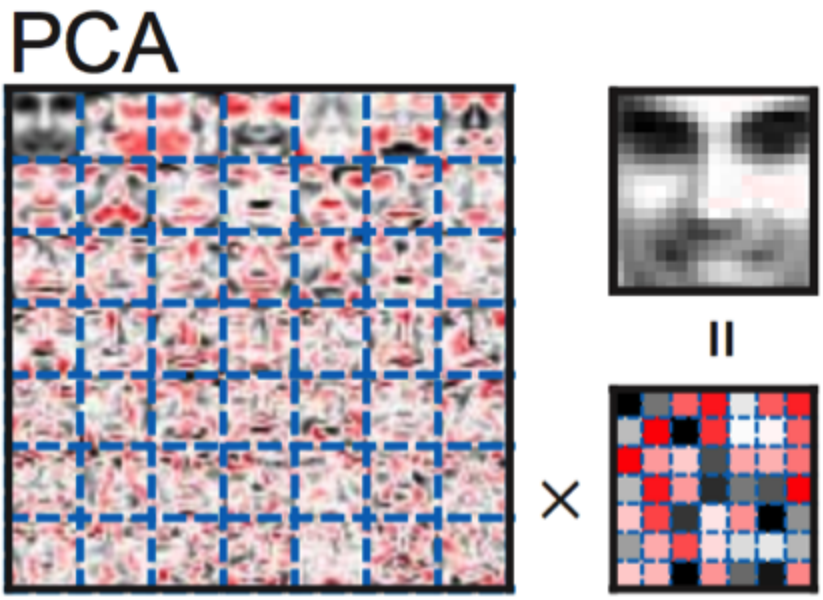
$$subject\ to\ W_{ij} \geq 0,\ H_{ij} \geq 0$$

NMF: Discussion on Image Dataset

When used on non-negative data such as images, NMF learns a parts-based representation of the dataset, resulting in interpretable models; while PCA learns the holistic representation. Example of facial recognition:
Basis Image given by NMF



Basis Image given by PCA



Here positive values are denoted with black pixels
and negative values with red pixels.

All three algorithms (PCA, ICA and NMF) learn to represent a face using linear combination of basis images. We see that NMF picks out individual components like nose, eyes and mouth, which corresponds well with our intuitive notion of faces. The reason for this is that first, only additive combinations are allowed in NMF and so this is compatible with the intuitive notion of combining parts to form a whole. Second, the basis images are sparse (It is an advantage because they are non-global and should contain several versions of mouths, noses etc. The variability of the face is generated by combining these different parts). PCA, on the other hand, gives out noisy components that doesn’t offer much interpretability. The reason for this is because first, PCA lets each face to be approximated by a linear combination of all the basis images (the vectors aren't sparse) and secondly, it allows the entries of the factorized vectors to be of arbitrary sign. And since these generally involve complex cancellations between positive and negative numbers, many basis images will lack intuitive meaning. ICA basis images (which isn's shown here), are independent holistic representations. The independence assumption made by ICA is ill-

suited for learning parts-based representation because various parts are likely to occur together.

This is a biconvex optimization problem (convex in H when W fixed, convex in W when H fixed). For implementation, we used the nnmf command in Matlab (this algorithm uses Alternating Least Squares for solving).

NMF: Other Application

More generally, NMF models the directly observable variables from the hidden variables, where each hidden variable activates a subset of visible variables ("part"). Activation of a collection of hidden variables combine these parts additively to generate a whole. Seen from this perspective, another often used application of NMF is semantic analysis of text document (think text mining).

Note: Reference: "Learning the Parts of Objects by Nonnegative Matrix Factorization", D.D.Lee, H.S.Seung.

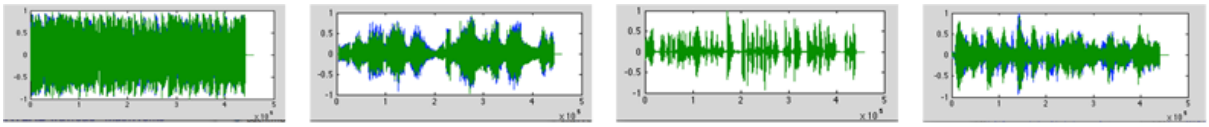
Application and Result
Comparisons of two unsupervised machine learning applications using PCA, ICA, and NMF.

We used these three algorithms for two interesting applications: Blind Source Separation (Cocktail Party Problem) and Hand-written Digit Recognition.

Blind Source Separation (Cocktail Party Problem)

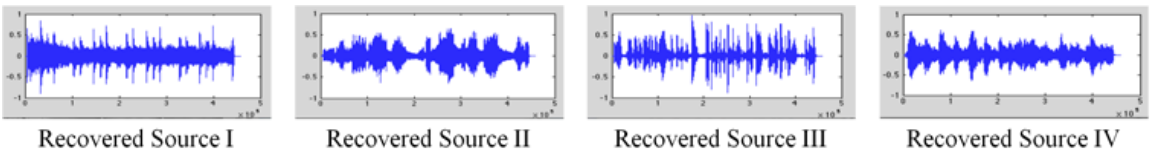
Imagine yourself at a cocktail party, and Haley is telling a boring story. You are much more interested in the gossip that Alex is telling Sam, so you tune out Alex and focus on Sam's words. Congratulations: you have just demonstrated the human ability to solve the "cocktail party problem" — to pick out one thread of speech from the babble of two or more people. Here, we use machine learning techniques to achieve this goal. In this example, we have four different mixtures of four different sounds (CNN new material, Bach's Symphony, Finnish Song and Pop Song Breakeven), mimicking the effect of having four microphones at four different places in a room (The number of mixtures should be equal to or greater than the number of sound sources for these algorithms to work). We feed this mixed data matrix (8 components in total, two channels for each audio file) into ICA, and the four components with the largest singular values corresponds to the four components we are looking for. The waveforms are plotted below. There's little noise in the background, but it's still very identifiable. PCA and NMF, however, only pick out one component.

Source Waveforms



Source I | Source II | Source III | Source IV

Recovered Waveforms using fastICA

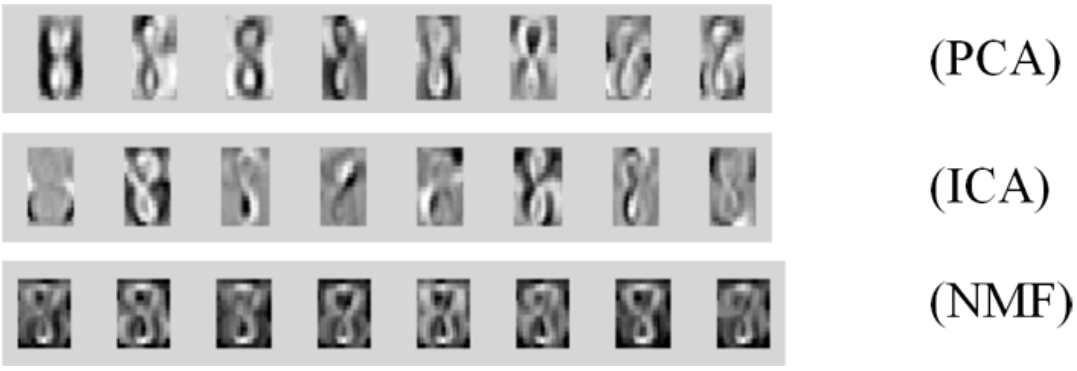


Another extended interesting feature: If we have two different mixtures of the two channels in a song, ICA will be able to extract the background music from a song (think karaoke).

Hand-written Digit Recognition

Have you ever thought about how USPS handles millions of mails each day? Obviously it will be great pain and tremendous work if postmen have to identify the postcodes people scribble manually and sort them into different bins. Their solution is to scan the handwritten addresses and utilize machine learning techniques to achieve automatic sorting. Here we ask our algorithms to learn the most significant features of digit 8. The dataset contains 542 samples. Here's some example digits in the dataset:

and the first eight components given by PCA, ICA and NMF:



The interpretation for these components is as follows: the first component captures the most significant feature of digit 8, meaning that most of the 8 in the dataset have this feature (looks like this), while the succeeding ones gives the second most significant feature etc. We see that NMF picks out the most significant features of 8 in its first component, compared to ICA

and NMF. It is a little hard to characterize the result besides just looking at it. If we do have labels (supervised learning), and turn this into a classification problem in which we may have more than one digits, the result can be more accurately characterized in terms of training error/test error.

Conclusion

A brief comparison of specialties of PCA, ICA, and NMF.

From the intuition behind the mathematical formulation and the results given by our chosen dataset, we reach the conclusion that different matrix factorization techniques work well towards different ends:

NMF works well for modeling non-negative data such as images. It finds sparse and parts-based representation of the data.

ICA works well for finding independent sources when the data isn’t Gaussian.

PCA has much broader application than ICA and NMF; it is ideal for pattern recognition and dimension reduction.

Future Work
Potential Future developments on the project.

Future Work

There are extensions to these popular matrix factorization techniques. We want to try these different matrix factorization techniques on different datasets in the future.

When the dataset is nonlinear, kernel trick (it does a nonlinear mapping from the original space to an inner product space so that the observations will gain meaningful linear structure in the new space) can be combined with PCA to achieve a non-linear dimensionality reduction (called Kernel PCA).

In some cases, if we want a sparse set of coefficients (weights in the linear combination of variables equal zero), Sparse PCA can help us with that.

In other cases, if we want to find clusters where columns and rows can be grouped together as a cluster (think bi-clustering), two-way PCA can be helpful.

Code
code
Matlab Code

```
% Blind Source Separation

% Four sounds are mixed together: (6 sec in total)
% - Bach's Brandenburger concerto No.6, Allegro
% - CNN News Material
% - Breakeven
% - Finnish Christmas Song: "A Sparrow on Christmas Morning"

% Here we have four different mixing sounds samples. By
performing ICA on
% them, we can successfully recover each individual sound.

if isempty(which('fastica'))
    display('add path of FastICA toolbox');
    addpath(strcat(pwd,'\FastICA_25'));
end

% read sound files in
[x1,Fs] = audioread('Bach_mixed.wav');  % source #1 sound track
[x2,Fs] = audioread('CNN_mixed.wav',[1 size(x1,1)]);  % source
#2 sound track
[x3,Fs] = audioread('Finnish_mixed.wav',[1 size(x1,1)]);  %
source #3 track
[x4,Fs] = audioread('Spanish_mixed.wav',[1 size(x1,1)]);  %
source #4 track


t = linspace(0,size(x1,1)/Fs,size(x1,1));  % time axis


% ICA analysis using FastICA
x = [x1,x2,x3,x4]';
r = fastica(x,'g','gauss');


% The output levels of this algorithm are arbitrary, normalize
to 1
r = r/max(max(abs(r)));

% save output audio file
audiowrite('PC1_ica.wav',r(1,:),Fs);
audiowrite('PC2_ica.wav',r(2,:),Fs);
audiowrite('PC3_ica.wav',r(3,:),Fs);
```

```
audiowrite('PC4_ica.wav',r(4,:),Fs);


% comparison with Nonnegative Matrix Factorization
[W,H] = nmf(x,8);
H = H/max(max(abs(H)));

audiowrite('PC1_nmf.wav',H(1,:),Fs);
audiowrite('PC2_nmf.wav',H(2,:),Fs);
audiowrite('PC3_nmf.wav',H(3,:),Fs);
audiowrite('PC4_nmf.wav',H(4,:),Fs);


% comparison with PCA
[U,D,V] = svds(x,8);
V = V/max(max(abs(V)));   %normalize

audiowrite('PC1_pca.wav',V(:,1),Fs);
audiowrite('PC2_pca.wav',V(:,2),Fs);
audiowrite('PC3_pca.wav',V(:,3),Fs);
audiowrite('PC4_pca.wav',V(:,4),Fs);




% Handwritten Digit Recognition
% recognize number 8

data = csvread('train8.txt');

% plotting some images
colormap('gray')
for i=1:12
    subplot(3,4,i)
    mat = flipud(rot90(reshape(data(i,:),16,16)));
    imagesc(mat)
    axis off
end


figure(3)
%NMF
addpath('~/matlab/nmf_toolbox_ver1.4')
[W,H] = nmf(zscore(data),12);
colormap('gray')
for i=1:8
    subplot(3,8,i)
    mat = flipud(rot90(reshape(H(i,:),16,16)));
```

```
    imagesc(mat)
    axis off
end


%ICA
addpath('~/matlab/FastICA_2.5/FastICA_25')
[S,A,Wh] =
fastica(zscore(data),'lastEig',12,'g','tanh','maxNumIterations'
,1e6);
colormap('gray')
for i=1:8
    subplot(3,8,i+8)
    mat = flipud(rot90(reshape(S(i,:),16,16)));
    imagesc(mat)
    axis off
end


%PCA
[U,D,V] = svds(zscore(data),8);
colormap('gray')
for i=1:8
    subplot(3,8,i+16)
    mat = flipud(rot90(reshape(V(:,i),16,16)));
    imagesc(mat)
    axis off
end
```

Matlab Code for Blind Source Separation and Hand-written Digit Recognition

Poster
Poster

Comparing Different Matrix Factorization Techniques for Unsupervised Learning

Qijia Jiang[1], Boying Meng[1] and Xuyang Lu[1]

[1]Department of Electrical and Computer Engineering, Rice University, Houston, Texas, U.S.A

RICE Unconventional Wisdom

Introduction

In real life, often, we come across data that doesn't have outputs (labels). In these cases, we strive to find the hidden structure and latent groupings within the data. This is when unsupervised machine learning comes into play. Popular unsupervised learning techniques include K-means, SOM and various clustering algorithms (spectral clustering, hierarchical clustering). Here, we discuss unsupervised machine learning within the context of various matrix factorization techniques.

Motivation

- Let's assume we have a data matrix of size nxp, where n corresponds to n observations and p corresponds to p features; very often, p is very large and the data is noisy. What we want is to find the association among the variables and uncover the major patterns underlying the data.
- What matrix factorization does is to reduce the dimensionality – it projects data matrix into a lower dimension space so what seems random and messy in the high dimension becomes structured and grouped in lower dimension.

PCA

- Short for Principal Component Analysis – closely related to SVD and eigenvalue problem (Ridge Regression, also)
- First PC (Principal Component) – direction that maximizes the sample variance
- Succeeding ones – direction that gives highest variance under the constraint of it being orthogonal (uncorrelated) to preceding ones
- Model looks like this:

$$\underset{U,D,V}{\text{minimize}} \frac{1}{2}\|X - UDV^T\|$$
$$\text{subject to } U^TU = I, \quad V^TV = I, \quad D \in diag^+$$

- Interpretation (pattern recognition): first column of U gives the first major pattern in sample (row) space; first column of V gives the first major pattern in feature (column) space
- Essentially, we are doing a coordinate transformation and the newly formed coordinate axes are the major patterns we are looking for
- Really easy to implement: feed the data matrix(n*p) to the SVD command in Matlab, extract the PC loading(V) and PC score(U) vector

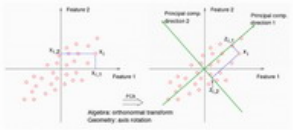


Figure Credit: https://radimrehurek.com/gensim/tut2.html#id3

ICA

- Short for Independent Component Analysis
- Imagine the data we have is a linear mixture of unknown latent factors and the factors are mutually independent and non-Gaussian, our goal is to identify these components
- Weakness of PCA: assumes Gaussianity – the condition $U^TU = I$ and $V^TV = I$ imply independence only when the data is Gaussian; what ICA does is to find statistically independent (rather than uncorrelated) sources
- Model:

$$X = AS$$
$$\hat{S} = W\hat{X}$$

  - where A is the mixing matrix and S is the source signals (rows of S independent)
  - We recover the signal using the "whitening" matrix W (where $W = A^{-1}$)
- Algorithms: Entropy-based, Neg-Entropy (Gaussian has maximal entropy, we want it to be as far from Gaussian as possible)
- What we use: FastICA[2] algorithm converges quickly

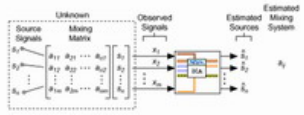


Figure Credit: http://cnx.org/content/m12495/latest/?collection=col10284/1.2

NMF

- Short for Nonnegative Matrix Factorization
- Solves another weakness of PCA: PCA tends to group both positively correlated & negatively correlated components together
- Model looks like this (for continuous data):

$$\min_{W,H} \frac{1}{2}\|X - WH\|_F^2$$
$$\text{subject to } W_{ij} \geq 0, H_{ij} \geq 0$$

- What NMF does: by forcing W and H to be positive, it finds patterns with same direction of correlation
- Another point of view: fuzzy clustering – W gives probabilistic cluster memberships, columns of H give variables that define each
- What we use: nnmf command in Matlab
- For count data, model looks like this:

$$\underset{W,H}{\text{maximize}} \sum_{ij}\left[X_{ij}\log(WH)_{ij} - (WH)_{ij}\right]$$
$$\text{subject to } W_{ij} \geq 0, H_{ij} \geq 0$$

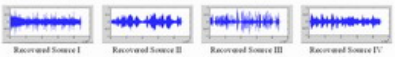
Result & Comparison

- **Handwritten Digits Dataset[3] (use digit 8 as example)**
  - Some example digits

  - Comparing PCA, ICA and NMF by plotting the first 8 components

  - For this dataset, we see that NMF performs better than PCA and ICA. It extracts the most significant features of 8.
- **Cocktail Party Problem (Blind Source Separation)**
  - Four different mixtures of four sources: Bach's Symphony, CNN News Material, Finnish Song and Pop Song Breakeven
  - Fed into FastICA algorithm, 8 components in total (two channels for each audio file), pick the largest four components
  - Original Waveforms

  - ICA works best in this case, it picks out all four sounds (with a little noise) while NMF and PCA only picks out one.
  - Recovered Waveforms using ICA (best result)



- **Interesting application: we succeed in extracting the background music in a song**

Conclusion

Different matrix factorization techniques work well towards different ends:

- NMF works well for modeling non-negative data such as images
- ICA works well for finding independent sources when the data isn't Gaussian
- PCA has much broader application than ICA and NMF; it is ideal for pattern recognition and dimension reduction

Reference

[1] Trevor Hastie, Robert Tibshirani, Jerome Friedman, "The Elements of Statistical Learning", Springer, Second Edition

[2] Hyvärinen, A (1999). *Fast and Robust Fixed-point Algorithms for Independent Component Analysis*. IEEE Transactions on Neural Networks, 10(3):626-634

[3] Normalized Handwritten Digits Dataset. Le Cun et al., 1990. AT&T Research Labs

Contact Info: Jiang (qj2@rice.edu), Meng (bm15@rice.edu), Lu (xl27@rice.edu)